2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Towards Deep Learning Assisted Autonomous UAVs for Manipulation Tasks in GPS-Denied Environments

Ashish Kumar†, Mohit Vohra†, Ravi Prakash†, L. Behera†, *Senior Member IEEE*
{https://github.com/ashishkumar822, https://youtu.be/kxg9xmr3aEM}

***Abstract*— In this work, we present a pragmatic approach to enable unmanned aerial vehicle (UAVs) to autonomously perform highly complicated tasks of object pick and place. This paper is largely inspired by challenge-2 of MBZIRC 2020 and is primarily focused on the task of assembling large 3D structures in outdoors and GPS-denied environments. Primary contributions of this system are: (*i*) a novel computationally efficient deep learning based unified multi-task visual perception system for target localization, part segmentation, and tracking, (*ii*) a novel deep learning based grasp state estimation, (*iii*) a retracting electromagnetic gripper design, (*iv*) a remote computing approach which exploits state-of-the-art MIMO based high speed (5000Mb/s) wireless links to allow the UAVs to execute compute intensive tasks on remote high end compute servers, and (*v*) system integration in which several system components are weaved together in order to develop an optimized software stack. We use DJI Matrice-600 Pro, a hexrotor UAV and interface it with the custom designed gripper. Our framework is deployed on the specified UAV in order to report the performance analysis of the individual modules. Apart from the manipulation system, we also highlight several hidden challenges associated with the UAVs in this context.**

I. Introduction

Despite being an easy task for humans, object manipulation using robots (robotic manipulation) is not very straight forward. Robotic manipulation using multi-DoF robotic arms has been studied extensively in the literature and has witnessed major breakthrough in the past decade, thanks to the advent of deep learning based visual perception methods and high performance parallel computing hardware (GPUs, TPUs). Various international level robotics challenges such as DARPA, Amazon Picking 2016 and Robotics Challenge-2017, have played a major role in pushing state-of-the-art in this area.

Autonomous robotic manipulation using UAVs, on the other hand, has just started marking its presence. It is because, developing low cost UAVs or vertical-takeoff-landing (VTOL) micro-aerial-vehicles (MAV)[1] has only recently become possible. The low cost revolution is primarily driven by the drone manufacturers such as DJI which covers $\sim 70\%$ of the drone market worldwide and provides low cost industrial drones for manual operation. Apart from that, the open-source autopilot projects such as Ardupilot [1] have also contributed in this revolution. The above reasons have resulted in their increased popularity among the worldwide researches

†All authors are with the Department of Electrical Engineering, Indian Institute of Technology, Kanpur {krashish,mvohra,ravipr,lbehera}@iitk.ac.in

[1]used interchangeably with UAV throughout the paper

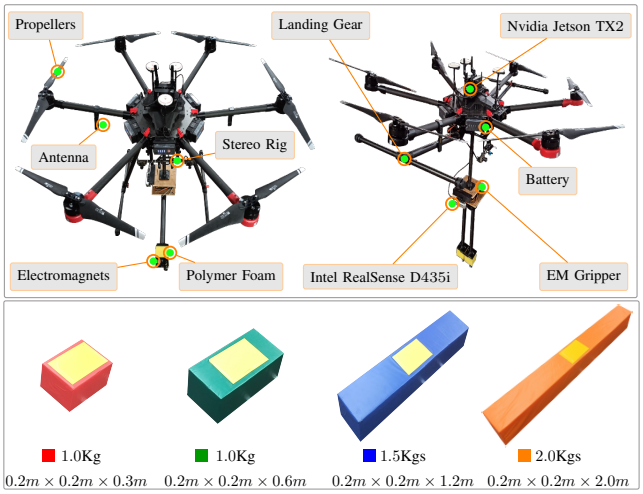


Fig. 1: Top: M600 Pro with our gripper. Bottom: Bricks

to develop algorithms for their autonomous operations. In the area of manipulation using UAVs, one of the most prominent and visible effort comes from E-commerce giant Amazon, a multi-rotor UAV based delivery system.

Another famous example is Mohammad Bin Zayed International Robotics Challenge (MBZIRC) 2020, which is establishing new benchmarks for the drone based autonomy. The Challenge-2 of MBZIRC 2020 requires a team of UAVs and an Unmanned Ground Vehicle (UGV) to locate, pick, transport and assemble fairly large cuboidal shaped objects (bricks (Fig.1)) into a given pattern to unveil tall 3D structures. The bricks consist of identifiable ferromagnetic regions with yellow shade.

Inspired by the challenge discussed above, in this paper, we present an end-to-end industry grade solution for UAV based maniulation tasks. The developed solution is not limited to constrained workspaces and can be readily deployed into real world applications. Before diving into the actual solution, we first uncover several key challenges associated with UAV based manipulation below.

A. Autonomous Control

Multi-rotor VTOL MAVs UAVs are often termed as under-actuated, highly non-linear and complex dynamical system. These characteristics allow them to enjoy high agility and ability to perform complex maneuvering tasks such as mid-air flipping, sudden sharp turns etc. However, the agility comes at a cost which is directly related to its highly

coupled control variables. Due to this reason, UAV control design in not an easy task. Hierarchical Proportional-Integral-Derivative (PID) control is one of the popular approaches which is generally employed for this purpose. Being quite simple, PID controllers are not intelligent enough to account for dynamic changes in the surroundings while their autonomous operations. To cope up with this, recently, the focus of researchers community has shifted towards machine learning based techniques for UAV control. However, due to data dependency and lack of generalization, learning based algorithms are still far from real deployable solution. Therefore, robust control of UAVs for autonomous operations still remains a challenge.

B. Rotor Draft

UAV rotor draft is a crucial factor which must be considered while performing drone based object manipulation. In order to execute a grasp operation, either the UAV must fly close to the target object or must have a long enough manipulator so that rotor draft neither disturbs the object nor the UAV itself. The former is only the case which is feasible but it's not an easy task. When flying at low altitudes, the rotor draft severely disturbs the UAV and therefore, poses a significant difficulty in front of stabilization and hovering algorithms. The latter, on the other hand, is not even possible as it would require a gripper of several meters long in order to diminish the effects of rotor draft. In addition, even a grasping mechanism $1-2m$ long, will increase the payload of UAV, resulting in quicker power source drainage. Also, such a long gripper will be infeasible to be accommodated into the UAV body.

C. Dynamic Payload

Dynamic payload attachment to the UAV is another important issue which arises after a successfull grasp operation. When a relatively heavier payload is attached to the UAV dynamically, the system characteristics and static thrust a.k.a hovering thrust also needs to be adjusted. However, as static thrust is increased in order to compensate for the gravity, the issues of heating of batteries and propulsion system arises. This raises safety concerns about the battery system and also decreases operational flight time. Although, addition of small weights to the UAV can be ignored, this case becomes severe for larger payload weights i.e. when the weight to be attached is order of 4 Kg while having an UAV with payload capacity 6Kg. Thus, requirement of intelligent control algorithms which can handle non-linearities associated with UAVs, becomes evident.

D. Visual Perception

In order to perform a seemingly easier task of picking, it requires a well defined sequence of certain visual perception tasks to be executed. This primarily includes object detection, instance detection and segmentation, instance selection, and instance tracking. While performing these tasks in constrained environments such as uniform background, the perception algorithms turns out be relatively simpler. However, in the scenarios such as outdoors, several uncontrolled variables jumps in, for example, ambient light, dynamic objects, highly complex and dynamic visual scenes, multiple confusing candidates. All such external variables increase the difficulty level to several degrees. Apart from that, real-time inference of perception modules is also desired, however, limited onboard computational and electric power further convolutes the system design.

E. Localization and Navigation

UAV localization and navigation play an important role in exploring the workspace autonomously and execute waypoint operations. In the GPS based systems, location information can be obtained easily and optionally fused with inertial-measurements-units (IMUs) for high frequency state-estimation. However, in GPS denied situations, the localization and navigation no longer remain an easy task from algorithmic point of view. In such scenarios, visual-SLAM or visual odometery based algorithms are generally employed. These algorithms perform feature matching in images, 3D point clouds and carryout several optimization procedures which in turn require significant amount of computational power, thus transforming the overall problem into a devil.

Due to the above mentioned complexities, limitations and constraints, performing the task of object manipulation using UAVs is not easy. Hence, in this work, we pave a way to solve the problem of UAV based manipulation in the presence of several above discussed challenges. In this paper, we primarily focus on real-time accurate visual perception, localization and navigation of using visual-SLAM for 6-DoF state-estimation in GPS-denied environments. The state-estimation is used for several other high level tasks such as autonomous take-off, landing, planning, navigation and control. Later, these modules are utilized to perform an object pick, estimation of successful grasp, transport and place operation which is primarily governed by our deep learning based visual perception pipeline. Highlights of the paper are as follows:

1) Identification and in detailed discussion of the challenges associated with UAV based object manipulation.
2) A novel computationally efficient, deep learning based unified multi-task visual perception system for instance level detection, segmentation, part segmentation, and tracking.
3) A novel visual learning based grasp state feedback.
4) A remote computing approach for UAVs.
5) An electromagnet based gripper design for UAVs.
6) Developing high precision 6-DoF state estimation on top of ORB-SLAM2 [2] visual-SLAM.

A side contributory goal of this paper is to introduce and benefit the community by providing fine details on low level complexities associated with UAV based manipulation and potential solutions to the problem. In the next section, we first discuss the available literature. In Sec. III, we discuss the system design. In Sec. IV, we discuss the unified visual perception system. In Sec. V, system integration is discussed.

The Sec. VI provides the experimental study and Finally, the Sec. VII provides the conclusions about the paper.

II. Related Work

Due to diverse literature on robotic vision and very limited space, we are bound to provide very short discussion on relevant work. We introduce only a few popular learning based approaches for the instance detection, segmentation, tracking and visual-SLAM. AlexNet [3], VGG [4], ResNet [5] are very first Convlutional Neural Networks (CNN). The object detectors Faster-RCNN [6], Fast-RCNN [7], RCNN [8] are developed on top of them. FCN [9], PSPNet [10], RefineNet [11] are approaches developed for segmentation tasks. Mask-RCNN [12] combines [6] and FPN [13] to improve object detction accuracies and also proposes an ROI align approach to facilicate instance segmentation.

ORB-SLAM [14] and ORB-SLAM2 [2] are popular approaches for monocular and stereo based visual-SLAM. UnDeepVO [15] is a very recent approach to visual odometery using unsupervised deep learning. DeepSORT [16] is a deep learning based multi object tracker inspired by kalman filter.

All of the above algorithms are being used in robotics worldwise and many recent works revolves around them. Nonetheless, the issue of deploying these algorithms on computationally limited platforms altogether is still a challenge.

III. System Design

A. UAV Platform

According to our experimental observation, an MAV must have a payload capacity atleast double of the maximum payload to be lifted. It is required in order to avoid complexities involved with dynamic payload attachment (Sec. I-C). Keeping this observation in mind, we use DJI Matrice-600 Pro hex rotor (Fig. 1) which has a payload capacity of 6 Kgs, flight time of 16 and 32 minutes with and without payload respectively. The UAV flight controller can be accessed through a UART bus via DJI onboard-SDK APIs. The DJI UAVs are quite popular worldwide for manual flying, however, turning them into autonomous platforms is not straight forward. It is because, being an industrial drone, it is optimized for manual control. Moreover, the SDK APIs lack proper documentation and other important details of low level control, the rate at which the controller can listen commands remain hidden. Availability of these details are quite crucial for robust autonomous control of the UAVs, since a delay of merely 20ms in the control commands can degrade the performance severely. Despite the challenges, their low cost and decent payload capacity make them attractive choice for research. It encourages us to examine the control responses, delays associated in the communication system of the UAV in order to adapt it for autonomous operations.

B. Gripper Design

We develop a retractable servo actuated gripper (Fig. 2) enabled with electromagnetic grasping. It consists of two carbon fiber tubes, called left and right arms. Robotics grade heavy duty servos are employed to arm and disarm the gripper when in air. Four servos, two on each side are used for high torque and better design stability. Two electromagnets (EMs) on each arm are used for the gripping mechanism. Each arm and respective electromagnets are connected via an assembly which consists of polymer foams sandwiched between carbon fiber plates. The carbon fiber tubes are linked at both ends (servo and Foam assembly) via 3D printed parts as shown in Fig. 2. While performing a grasping operation, the foam compression/decompression action accounts for the drifts in UAV's actual and desired height. The control circuit and firmware is developed on PIC18F2550, a Microchip USB-series microcontroller.

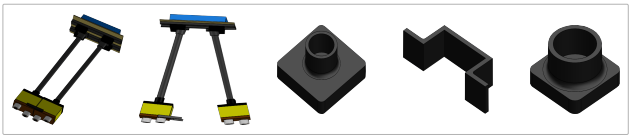


Fig. 2: The CAD design of our gripper and 3D printed parts

C. Compute Infrastructure and Communication Device

We equip our platform with DJI-Manifold 2 mini computer, based on NVIDIA Jetson Tegra TX2. The computer is powered by 6 core CPUs along with 256 GPU cores, 8GB RAM and 32GB eMMC. A dual band 150 (2.4GHz) + 717 (5GHz) = 867Mbps onboard WiFi is also available. We use MIMO based ASUS ROG RAPTURE GT-AX-11000 Tri-Band wireless router, in order to link base station computers with onboard computer. The router offers very high data transfer rates of upto 1148Mbps on 2.4GHz and 4802Mbps on two 5GHz bands, aggregating upto 11000Mbps. The motivation behind using such high speed device is to enable real-time high performance remote computing, system monitoring, remote data logging, and algorithmic debugging. The router also make it feasible to develop multiple UAVs based solutions.

IV. Unified Multi-Task Visual Perception

Deep learning based visual perception systems perform with very high accuracy on tasks such as object detection, instance detection, segmentation, visual tracking. All of these methods exploit the power of CNNs to generate unique high dimensional latent representation of the data. During the past decade, a number of CNN based algorithms have been proposed which have shown increasing improvements in these tasks over time. The CNN models of these algorithms are well tested on standard datasets, however, lacks generalization. These models, often, are also very large in size. Therefore, direct deployment of such algorithms for real-time robotic applications is not a wise choice.

As an example of our case, four tasks i.e. instance detection, segmentation, part segmentation and object tracking need to be performed in order to execute a high level task. It is quite important to note that despite the availability of several algorithms and their open source implementations

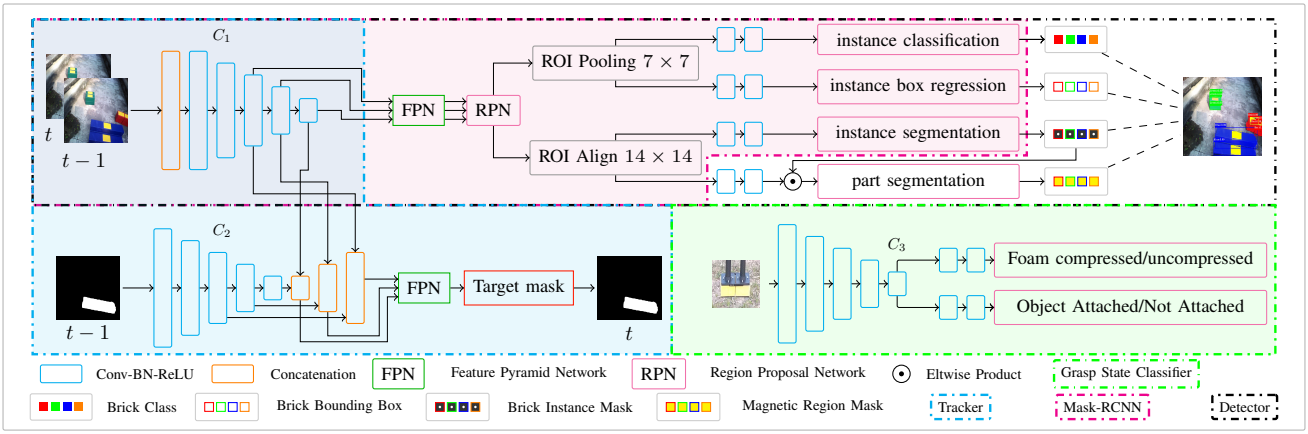


Fig. 3: Unified Visual Perception System

individually, none of them combines these primary tasks together.

Therefore, in the realm of limited computational and power resources, we put an effort to combine aforementioned four tasks in a unified CNN powered visual perception framework. Instead of developing perception modules from scratch, we develop the unified multi-task visual perception system on the top of the state-of-art algorithms. It is done in order to achieve the goals of computational and resource efficiency, real-time, robust and fail-safe performance.

Being a highly complex visual perception pipeline, we try our best to explain the overall perception system pictorially in Fig. 3. Below we have discussed each of the component in detail.

A. Instance Detection and Segmentation

In the presence of multiple instances of an object, it is necessary to perform instance level detection and segmentation. Mask-RCNN [12] is one of the popular choices for this purpose. The pretrained models of Mask-RCNN utilize ResNet-50, ResNet-101 as CNN backbones for large datasets such as MS-COCO, PASCAL-VOC. Run time performance of this algorithm is limited to 2-5 FPS even on a high-end GPU device having $\sim$ 3500 GPU cores. Hence, deploying even a baseline version of Mask-RCNN on Jetson TX2 category boards appears as a major bottleneck in algorithmic design.

As a solution to adapt Mask-RCNN for our system, we carefully develop an AlexNet style five stage CNN backbone with MobileNet-v1 style depthwise spearable convolution. After several refinements of parameter selection, we come up with a very small and lightweight model which can run in real-time. In the baseline Mask-RCNN, object detection is performed upto stage-2 of ResNet. However, due to resource limitation, we limit the object detection upto stage-3 starting from stage-5. Rest of the architecture for instance detection, segmentation remains same.

Further, in order to improve the accuracy, we first train the final model on mini ImageNet ILSVRC-2012 dataset so that primary layers of the model can learn meaningfull edge and color responses similar to primary visual cortex in biological vision system. We then fine tune the model for our task of instance detection and segmentation. Pretraining on ImageNet improves the accuracy, generalization and suppresses false positives.

B. Part Detection

The Mask-RCNN is only limited to the task of instance detection and segmentation. However, we have an additional requirement to localize specific part of an object, in our case, the ferromagnetic regions. Therefore, we extend the previously developed CNN infrastructure to accommodate this functionality. In order to achieve that, features from ROI-Align layer are passed through a stack of two convolution layers, similar to instance segmentation branch (Fig. 3). The features corresponding to an instance are then multiplied elementwise with the segmentation mask of the same instance obtained from instance segmentation branch. Mask-RCNN follows the object centric approach for instance segmentation and performs binary classification using binary cross entropy loss. We also follow the same approach to learn binary mask. For more detailed explanation, please refer to [12].

C. Target and Conditional Tracking and Visual Servoing

Once the instances of desired class are detected and segmented, the instance requiring minimal translation control efforts is selected for grasping operation. A binary mask M_{t-1} corresponding to the selected target is sent to the tracker. The overall tracking process is conditioned on M_{t-1} where the non-zeros pixels of M_{t-1} guides the tracker "where to look". For this reason, we term the target mask as support mask. The task of tracker thus can be defined as to predict support mask image M_t at current time step given current I_t and previous I_{t-1} RGB frames and support mask M_{t-1}.

We further extend the developed perception system so far to be used as lightweight tracker (Fig. 3). In order to realize the the tracking process, we design a separate AlexNet style CNN (C_2) (similar to C_1 and) to extract

spatial feature embeddings of the support mask image. The backbone C_2 has negligible number of weights as compared to the backbone C_1 for RGB images. We have designed the architecture such that both the tracker and the detector share a common CNN backbone C_1 for high dimensional embedding of RGB images. This is done in order to prevent the computational resources (limited GPU memory) from being exhausted.

Next, feature embedding of both RGB images and the support mask are fused by first performing a concatenation operation on the embeddings of stage-3, stage-4 and stage-5 of both C_1 and C_2. This step essentially represents the conditioning of the tracking process onto M_{t-1}. Later, these fused representations are aggregated by using FPN in order to predict highly detailed support mask for next time step. Despite the very small size of CNN architecture, the FPN module is responsible [17] for highly detailed support mask. Higher resolution embeddings are avoided due to overly large memory requirements.

To begin with the tracking process, first an instance in an image I_{t-1} is selected and a corresponding binary mask M_{t-1} is obtained. Later, I_t, I_{t-1} and M_{t-1} are fed to their respective CNNs C_1, C_2 of the tracker. A binary mask M_t is predicted by the tracker which depicts the location of the target object at time instant t. This newly estimated location of the target instance is then sent to the control system which perform real time visual servoing in order to execute a pick operation.

Further, it is important to note that while tracking, the detection modules following the FPN block of the detector (Fig. 3) are halted to save computations. The detection is generally performed on a single Image while tracking process requires two images to be fed to C_1. As mentioned previously, the detector and tracker shares a common backbone, therefore C_1 must be provided with two images regardless of the task. Therefore, I_{t-1} and I_t are essentially a copy of each other in the detection while these two are different during the tracking process.

D. Learning Based Feedback for Robust Grasping

In order to execute a robust grasping operation, continuous feedback of the grasp state i.e. "gripper contact with object", "object attached with gripper" are required. The feedback device typically depends on the nature of gripper. For example, finger gripper generally use force sensors to sense the grasp state. In our case, EMs are used which requires dedicated circuitry to obtain the feedbacks such as "whether the EMs are in contact with any ferromagnetic material". It poses additional design constraints and system gets extremely complicated.

To simplify the design, we translate the problem of obtaining grasp state feedback into image classification problem. In order to achieve that we take advantage of the fact that the foam assembly of our gripper gets compressed at the moment of contact with the object. In order to realize the approach, We develop a five stage CNN C_3 (Fig. 2) where the stage-5 is followed by two classification heads. One of them classifies an image into G_1 or G_2 whereas another classifies into G_3 or G_4.

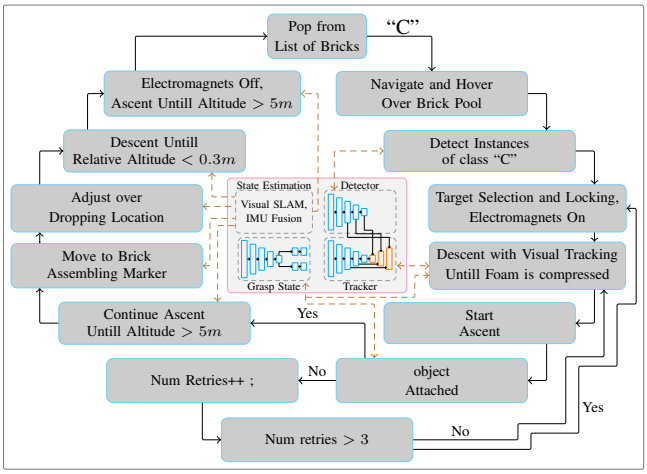


Fig. 4: Simplified State Machine of the overall System

The four classes G_1, G_2, G_3, G_4 represent the states "Foam Compressed", "Foam Uncompressed", "Object Attached" and "Object Not-Attached" respectively. The input image is captured from the downward facing Intel realsense D435i mounted on the gripper. Due to the fixed rigid body transformation between gripper and camera, the gripper remains visible at a constant place in the image. Therefore, input to the classifier is a cropped image. The cropping is done such that foam, the electromagnets and the ferromagnetic regions (in case object is attached) remains image centered. The cropped image is resized to 64×64 before feeding to the classifier. For the data collection and training details are provided in Sec. VI.

At the time of performing grasp operation, the UAV descents at very low speed with the help of visual servoing until the grasp state classifier predicts G_2, also referred as touch down. Once the touch down is received, the UAV starts ascending slowly while the grasp state classifier continuously monitors the gripper in this phase. Once the classifier confirms successfull grasp which is denoted by G_3, the UAV continues to ascent, transports and places the brick at desired location.

V. System Integration

A. Sensor Fusion and State Estimation

In order to navigate and perform various tasks accurately in a workspace, the UAV must be aware of its 6D pose i.e. $\{x, y, z, r, p, y\}$ at every instant of time. To address this issue in GPS-Denied environments, we develop a state-estimation solution which can provide a real-time state-estimation despite limited computational resources onboard the UAV.

The solution is based on very popular real time ORB-SLAM2 [2] algorithm. ORB-SLAM2 can work on stereo as well as RGB-D sensors. We prefer to use stereo based SLAM over RGB-D SLAM because of three reasons. First, RGB-D sensors quite costly as compared to RGB cameras and most

of them does not work in outdoor. Second, RGB-D sensors have several points with missing depth information. Third, RGB-D cameras have limited depth range. On the other hand, in stereo based SLAM, the depth of even far points can be computed provided adequate baseline separation between cameras. However, the accuracy of stereo based SLAM comes at a cost which is related to time consumed in keypoint detection, stereo matching, bundle adjustment procedure. The standard implementation of ORB-SLAM2 can process images of resolution 320×240 at 25 FPS on a core $i7$ CPU. When it is deployed onto the Jetson-TX2 along with other processes, the speed drops to 5FPS.

With the limited power budget, it is not possible to place another system alongside the TX2 board. Therefore, in order to solve this issue, we take advantage of our high speed networking infrastructure. In order to execute the SLAM algorithm, the captured stereo images are sent to a base station. The SLAM algorithm is executed on the received images to update the 6D UAV state and 3D map. As soon as 6D pose from SLAM is obtained, it is fused with IMU by using Extended Kalman Filter in order to obtain a high frequency state estimate.

B. Path Planning and Collision Avoidance

We use RRTConnect* algorithm for real time planning and flexible collision library (FCL) library for collision avoidance. In order to use them, we modify an open source project MoveIt!, which is was originally developed for robotic manipulators and already has support for RRTConnect* and FCL. However, MoveIt! has several bugs related to planning of 6D joints[2]. For our use, we resolve the issues and configure MoveIt! for UAV.

C. Control

As mentioned previously, DJI SDK APIs does not expose its internal controller details. It also doesn't allow the state-estimation sent to its internal controllers. Therefore, We tune four outer loop PIDs, one for each of vertical velocity, roll and pitch angle and yaw velocity respectively. Based on state-estimation, the tuned PIDs performs position control of the UAV by producing commands for internal hidden controllers,

Fig. 4 shows simplified state machine of our UAV based manipulation system. For the sake of brevity, very obvious steps have been dropped from the state machine. It can be noticed that the state machine can be completely explained by the visual perception, state-estimation, planning and control algorithms discussed previously.

VI. Experiments

In order to justify the validity and usefulness of our work, we provide the experimental analysis of the timing performance and accuracy of detector, tracker and grasp state classifier in the context of real-time autonomous robotic applications. Performance evaluation of the visual perception system on the public benchmarks for object detection such as MS-COCO, PASCAL-VOC is out of the scope of this paper.

[2]"Floating Joint" in the context of MoveIt!

A. Dataset

*1) **Detection**:* The data collection process is based on our work [17] which won 3rd prize in Amazon Robotics Challenge, 2017. Following [17], we collect a very small sized dataset of about 100 images which contains multiple instances of all four categories. We split the dataset into training and testing set in the ratio of $7:3$. We perform both box regression and segmentation for the instances while only mask segmentation is performed for the ferromagnetic regions. In order to achieve that, for each instance in an image, two masks are generated manually, one for instance itself and another for its ferromagnetic region. These masks serve as groundtruth for instance segmentation and part segmentation pipeline. The ground truth for box regression is extracted automatically based on the convex hull of the mask pixels belonging to the instance. The instance class-ID is defined at the time of mask generation. In order to multiply the dataset size, run time occlusion aware scene synthesis is performed similar to [17]. The synthetic scene synthesis technique allows us to annotate only a few images and appears to be a powerful tool for data multiplication.

*2) **Tracking**:* We collect 10 indoor and 10 outdoor video sequences at $\sim$ 30FPS with instance size varying from smaller to larger. We downsample the video sequences to $\sim$ 2 FPS and annotate each frame for instance detection, classification, segmentation and part segmentation. The overall dataset for tracking purpose roughly consists of 200 images. The 100 indoor images are used for training and 100 outdoor images are kept for testing. This is done in order to examine the generalization of the tracker. Synthetic scene generation [17] plays an important role in this process.

*3) **Grasp State Classification**:* In order to train the grasp state classifier, we collect four kinds of images (50 each).

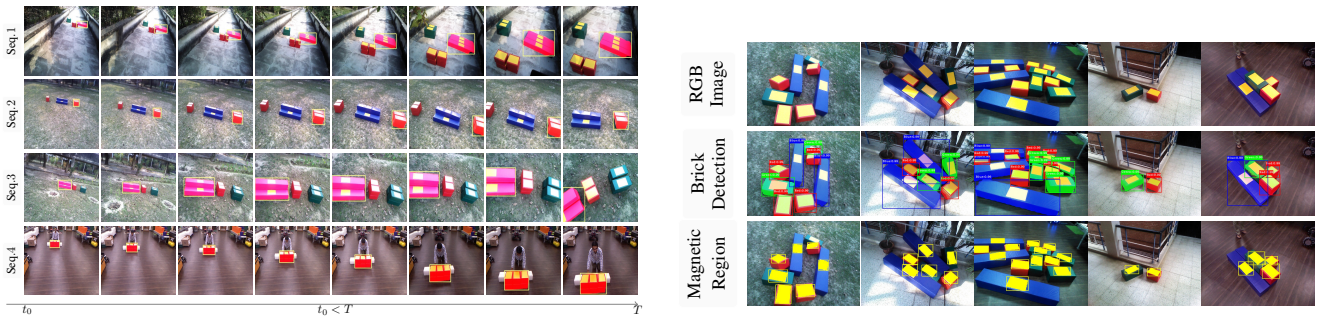
1) The foam in compressed state with object attached.
2) The foam in compressed state without object attached.
3) The foam in uncompressed state with object attached.
4) The foam in uncompressed state with no object.

The sets of images $\{1, 2\}$ and $\{3, 4\}$ represent the states G_1, G_2, G_3, G_4. All the collected images are cropped and resized as described in Sec.IV-D. Practically, the actual cropping region is kept slightly larger than discussed above. It is done in order to provide contextual details for better classification performance. The dataset is splitted into $2:3$ for training and testing i.e. 40 training and 60 test images are available for each category. While training, we again utilize synthetic scene synthesis to accommodate for the dynamic outdoor scenes. For more details, please refer to [17].

B. Training Policy

We refer the baseline architecture as Arch_1. We perform training and testing on three different variants Arch_1, Arch_2 and Arch_3. The Arch_2 and Arch_3 consists of 25% and 50% more number of parameteres as compared to Arch_1, in each layer of all CNN backbones. The kernel sizes for various layers have been provided in Table-I.

Since, detection and tracking are two different tasks, therefore, separate training is required. In order to avoid



(a) (b)

Fig. 5: Qualitative results for (a) Tracking, and (b) Instance detection, segmentation and part detection.

TABLE I: Kernel sizes. '$*$' equals to the input channels

Layer	C_1	C_2	C_3
Stage-1	$4 \times 3 \times 3 \times 3$	$2 \times 1 \times 3 \times 3$	$2 \times 3 \times 3 \times 3$
Stage-2	$8 \times 4 \times 3 \times 3$	$4 \times 2 \times 3 \times 3$	$4 \times 2 \times 3 \times 3$
Stage-3	$16 \times 8 \times 3 \times 3$	$4 \times 4 \times 3 \times 3$	$8 \times 4 \times 3 \times 3$
Stage-4	$32 \times 16 \times 3 \times 3$	$8 \times 4 \times 3 \times 3$	$8 \times 8 \times 3 \times 3$
Stage-5	$32 \times 32 \times 3 \times 3$	$16 \times 8 \times 3 \times 3$	$16 \times 8 \times 3 \times 3$
Others	$12 \times * \times 3 \times 3$	$4 \times * \times 3 \times 3$	$8 \times * \times 3 \times 3$

TABLE III: Performance Analysis of Perception System

Network Architecture	Time (ms)	Box mIoU	Seg mIoU	Part Seg mIoU	Tracker mIoU
Arch_1 (FP16)	35	80.3	79.2	73.1	83.4
Arch_1 (FP32)	69	82.2	80.0	72.9	84.8
Arch_2 (FP16)	41	81.2	79.7.	73.6	84.1
Arch_2 (FP32)	73	83.5	81.1	74.0	85.6
Arch_3 (FP16)	52	85.3	80.6	73.5	84.4
Arch_3 (FP32)	97	86.7	81.9	74.1	86.4

that, instances and parts are also annotated along with the masks in the training dataset of the tracker. In other words, the detector now can be trained on the tracker training data. To balance the training process, images are chosen randomly with a probability of 0.5 from both the detector and the tracker training dataset. Through this technique, the unified system experiences glimpse of temporal data (due to presence of video sequences) as well as ordinary single image object detection.

Further, we report mean-intersection-over-union (mIoU) scores for box detection, instance segmentation, and part segmentation. The mIoU score is a well known metric for reporting box regression and segmentation performance. Since, tracker essentially performs segmentation, therefore same is reported for the tracker. Due to very high inter-class and low intra-class variance, we did not encountered any misclassification of the bricks. Therefore, we have dropped the box classification accuracy from the results.

C. Training Hyperparameters

We use base learning rate = 0.01, learning rate policy=$step$, SGD optimizer with nestrov momentum = 0.9 for pre-training the backbone on mini imagenet dataset. While base learning rate = 0.001, learning rate policy=$step$, $ADAM$ optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ are used for finetuning on our dataset.

D. Ablation Study of Perception System

Table-III depicts the performance of various perception components along with the timing performance. For each performance metric, we perform ablation study of all the three architectures Arch_1, Arch_2, and Arch_3. It can be noticed that Arch_2 shows an minor improvement over Arch_1, however, Arch_3 does not show significant improvements despite an increment in the number of learnable parameters by 50%. In our views, It happens because the backbone architecture for all the three variants remains same except the representation power. Another reason for that is the objects under consideration are of uniform colours and therefore, intra-class variance is very low.

The table also shows various performance metrics for both Half-Precision (FP16) and single-precision (FP32) computations on GPU. A single precision floating point number consists of 32 bits whereas half precision consists of 16 bits. There also exists double precision FP64 which consists of 64 bits. As the number of bits of a floating point number increases, the smallest number representable decreases i.e. more details comes in. However, the amount of computation time required increases non-linearly and drastically. Therefore, in practice FP32 is used on desktop grade GPUs. However, due to real time constraints and power considerations, NVIDIA has recently developed half-precision based high-performance libraries especially for deep learning algorithms. We take advantage of half precision in our application without sacrificing the accuracy.

E. Synthetic Scenes and Comprehensive Data Augmentation

Tabel-II shows the effect of synthetic scenes and other augmentations as performed in [17]. It can be noticed that synthetic scenes alone contribute to the improved performance as compared to the other remaining augmentations (highlighted in Blue). The decreasing value of tracker mIoU is observed when blur is included in the augmentation process.

F. Unified vs Distributed Perception System

The unification of various tasks is a primary contribution of this paper. Hence, we compare the timing performance

TABLE II: Effect of Synthetic Scenes and Comprehensive Data Augmentation fro $Arch_1$, FP16

Augmentation						Box Detection	Segmentation	Part Segmentation	Tracker
colour	scale	mirror	blur	rotate	synthetic scenes	mIoU	mIoU	mIoU	mIoU
✗	✗	✗	✗	✗	✗	23.7	31.8	18.8	15.3
✓						25.3	32.1	21.3	17.7
✓	✓					30.1	38.6	29.1	23.1
✓	✓	✓				32.3	40.2	31.9	25.5
✓	✓	✓	✓			32.5	41.4	33.8	24.1
✓	✓	✓	✓	✓		37.2	49.8	37.7	28.4
✓	✓	✓	✓	✓	✓	80.3	79.2	73.1	83.4

TABLE IV: Unified Vs Distributed Perception System, $Arch_1$, FP16

Network Architecture	Time (ms)			
	Detection	Part Seg	Tracking	Total
$Arch_1$	−−	−−	−−	35
Detection	29	−−	−−	92
Part Segmentation	−−	30	−−	
Tracking	−−	−−	33	

TABLE V: Performance Analysis of Grasp State Classification

Network Architecture	Time (ms)	Foam State Accuracy(%)	Object Attached State Accuracy(%)
$Arch_1$ (FP16)	15	100.0	100.0
$Arch_1$ (FP32)	32	100.0	100.0
$Arch_2$ (FP16)	20	100.0	100.0
$Arch_2$ (FP32)	41	100.0	100.0
$Arch_3$ (FP16)	25	100.0	100.0
$Arch_3$ (FP32)	56	100.0	100.0

of the unified system against three different networks for detection, part segmentation and tracking (Table-IV). These isolated networks have exactly same configurations for each layer in CNNs and other required components such as RPN, FPN etc. It is clearly evident from the table that our unified pipeline is far better in terms of timing performance, because all the three modules can be run simultaneously. Fig. 5b and Fig. 5a show qualitative results of unified perception system. The results in Table-IV are with $Arch_1$ trained using all kinds of augmentations.

G. Grasp State Classification

The performance of grasp state feedback CNN is shown in Table-V. The definitions of $Arch_1$, $Arch_2$, and $Arch_3$ remains same, except the parameters of $Arch_1$ belong to the Grasp State Classifier C_3 (Fig. 3). The 100% accuracy is evident due to very simple classification task. Also, it is important to note that, for both $Arch_1$ and $Arch_2$, the timing performance of FP16 is roughly twice of FP32 where as it is not the case with $Arch_3$ (highlighted in red). It happens because of the non-linear and saturating speed-up curves of the GPU device.

VII. Conclusion

This work introduces an end-to-end framework for UAV based manipulations tasks in GPS-denied environments. A deep learning based real-time unified visual perception system is developed which combines the primary tasks of instance detection, segmentation, part segmentation and object tracking. The perception system can run at 30 FPS on Jetson TX2 board. A complete electromagnets based gripper design is proposed. A novel approach to handle the grasp state feedback is also developed in order to avoid external electronics. To the best of our knowledge, the unified vision system combining four tasks altogether and the grasp state feedback in the context of UAVs has not been developed in the literature. Apart from that, a remote computing based approach for 6DoF state-estimation is introduced. Certain improvements in the opensource framework MoveIt! to accommodate UAVs have also been done.

References

[1] A. D. Team, "Ardupilot," *URL: www. ardupilot. org, accessed*, vol. 2, p. 12, 2016.

[2] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[7] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[8] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "Cms-rcnn: contextual multi-scale region-based cnn for unconstrained face detection," in *Deep Learning for Biometrics*, pp. 57–79, Springer, 2017.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[10] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] G. Lin, A. Milan, C. Shen, and I. D. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation.," in *Cvpr*, vol. 1, p. 5, 2017.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *CVPR*, 2017.

[13] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *CVPR*, 2017.

[14] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[15] F. Liu, Z. Liu, and Q. Wu, "Monocular visual odometry using unsupervised deep learning," in *2019 Chinese Automation Congress (CAC)*, pp. 3274–3279, IEEE, 2019.

[16] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[17] A. Kumar and L. Behera, "Semi supervised deep quick instance detection and segmentation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8325–8331, IEEE, 2019.